MULTI-ROBOT PATROL VIA THE METROPOLIS-HASTINGS ALGORITHM

An Undergraduate Research Scholars Thesis

by

MATTHEW RYAN EDWARDS

Submitted to Honors and Undergraduate Research
Texas A&M University
in partial fulfillment of the requirements for the designation as an

UNDERGRADUATE RESEARCH SCHOLAR

Approved by
Research Advisor: Dr. Dylan Shell

May 2015

Major: Computer Science

TABLE OF CONTENTS

ABSTRACT

Multi-robot Patrol via the Metropolis-Hastings Algorithm. (May 2015)

Matthew Ryan Edwards
Department of Computer Science
Texas A&M University

Research Advisor: Dr. Dylan Shell
Department of Computer Science

The problem of multi-robot patrol is a growing field of study that focuses on the problem of coordinating teams of robots to optimally patrol a perimeter or area. In this paper, we propose a new method of generating patrolling policies in the form of Markov chains via the Metropolis-Hastings algorithm. Our proposed method generates non-deterministic patrolling policies with the purpose of minimizing the probability of adversarial attack to a given area. We compare our method to a wide variety of approaches to patrolling methods on a large set of graphs in order to test the effectiveness of Markov chains as a patrolling policy.

CHAPTER I

INTRODUCTION

To patrol is defined as the activity of going around or through an area at regular intervals for security purposes. In this context, patrolling should be performed by a team of robots. Multi-robot patrolling is a field of study which has been growing throughout the last decade [2, 3, 5]. Within this problem, researchers are searching for ways to optimally patrol an area or perimeter based on various factors including communication and coordination, environmental settings, and the presence of an adversary, in order to protect a designated area or set of valuable items.

Chevaleyre presents a paper that contains a theoretical analysis of the multi-robot patrolling problem [3]. The strategies analyzed fall into one of two categories: cyclic and partition-based strategies. The analysis of these strategies uses the concept of idleness, or length of time a node experiences between visits from the patroller, to determine the effectiveness of each strategy type. Through his work, Chevaleyre shows that the cyclic approach is more effective than partition-based strategies. Related to his paper is a survey by Almeida et al., which compares various approaches towards patrolling when utilizing an idleness criterion and provides high results for a cyclic approach [4].

Elmaliach et al. [2] presents frequency optimization criteria that can be used to evaluate patrolling policies, as well as an algorithm for generating a patrolling policy that guarantees maximal uniform, optimal frequency for all nodes on the graph. Their solution finds a circular

path that visits all points in an area while ensuring that patrolling robots are positioned uniformly along the path.

Agmon explores the problem of multi-robot perimeter patrol in the presence of an adversary [5]. Her work focuses on maximizing the chance of a patrolling robot detecting an adversary along different patrolling environments such as a perimeter or fence. Her experiments also utilize three different methods of patroller movement, and her resulting algorithms maximize this probability of detection for all movement models of the patrollers.

In another survey by Portugal et al. [1], multi-robot area patrolling algorithms are compared based on the criteria of robot perception, communication, coordination, and decision-making. This survey provides strengths and weaknesses for a variety of approaches to the multi-robot patrolling problem, ranging from randomized and partitioning algorithms to approaches using heuristics and the concept of idleness.

Objectives

In this project, we propose the generation of patrolling policies in the form of Markov chains with the use of the Metropolis-Hastings algorithm. The patrolling policies generated by this algorithm can be applied to any type of graph, as opposed to only area or perimeter graphs as in the examples above. Many patrolling policies that are frequency-based in their approach provide a deterministic solution which can be easily exploited by an adversary [5]. We theorize that by using a Markov Chain to represent a patrolling policy, the actions of a patroller will be much more difficult to predict by an adversary, thus making it more difficult for an adversary to attack an area undetected [6]. Through these experiments, we will explore different methods with

which to generate patrolling policies using the Metropolis-Hastings algorithm, and we will provide a comparison of our methods against existing patrolling schemes to test the usefulness of Markov chains as a potential representation of a patrolling strategy.

CHAPTER II

METHODOLOGY

Patrolling Setting

We define the patrolling environment as an undirected graph $G = (V, E)$ with a vertex set $V = \{1,\ldots,n\}$ and an edge set $E = \{1,\ldots,m\}$. Each vertex $v \in V$ corresponds to a region in the graph and each edge $e \in E$ to a connection between two regions in the graph. Let M be a discrete time Markov chain on the graph G where M_{ij} is the probability of transitioning from vertex i to vertex j. Using this formulation, a patroller can utilize the local information of its position on the graph to perform a weighted random walk. For instance, if the patroller is at vertex k, it utilizes the transition probabilities in the k-th row of M to weigh its random choice of which vertex to move to next (Figure 1). A robot using a Markov chain in this way can patrol a graph such that its decision-making and perception are independent and local to the agent from the other loosely coupled patrollers, thus bypassing many communication and coordination constraints [1].

When using Markov chains to patrol, the stationary distribution (π) becomes an important property. The stationary distribution of a Markov chain provides the long-run probabilities of how often one can expect to be in any state within a Markov chain. To illustrate this, suppose we have a two state Markov chain, as depicted in Figure 2. The stationary distribution of this Markov chain can be thought of as a weighted coin where heads represents state 1 and tails represents state 2. If we flip this coin, then there is some chance p of getting heads and some chance $1-p$ of getting tails. This analogy can be extended to Markov chains containing n states where the stationary distribution can be represented by a weighted n-sided object. In a patrolling

setting, the stationary distribution of a Markov chain strategy describes the probability that one can find a patroller in any given node. To use the example of the two state Markov chain again, if we flip the weighted coin, there is some probability *p* that the patroller will be in location 1 and some probability *1-p* that the patroller will be in location 2.

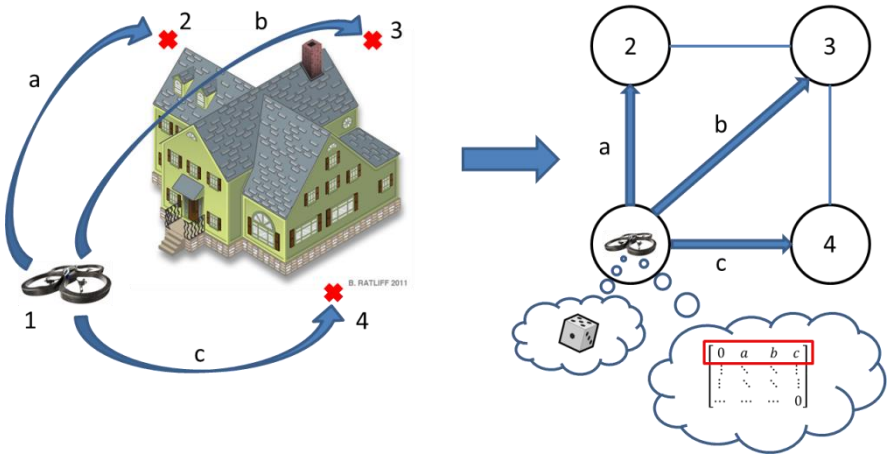

Figure 1: Choosing a Movement – On the left is the physical world visualization of the robot deciding where to patrol next. On the right is the abstraction of the environment. There, the robot uses the Markov chain for the graph to roll a weighted die (with probabilities in the red box) to decide the location it will move to next.

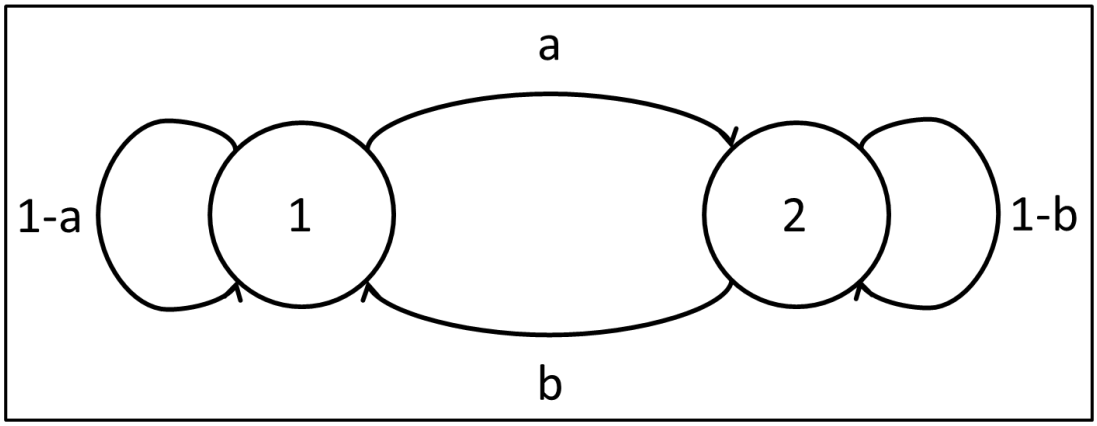

Figure 2: Two State Markov Chain – A two state Markov chain with transition probabilities represented as variables *a* and *b*.

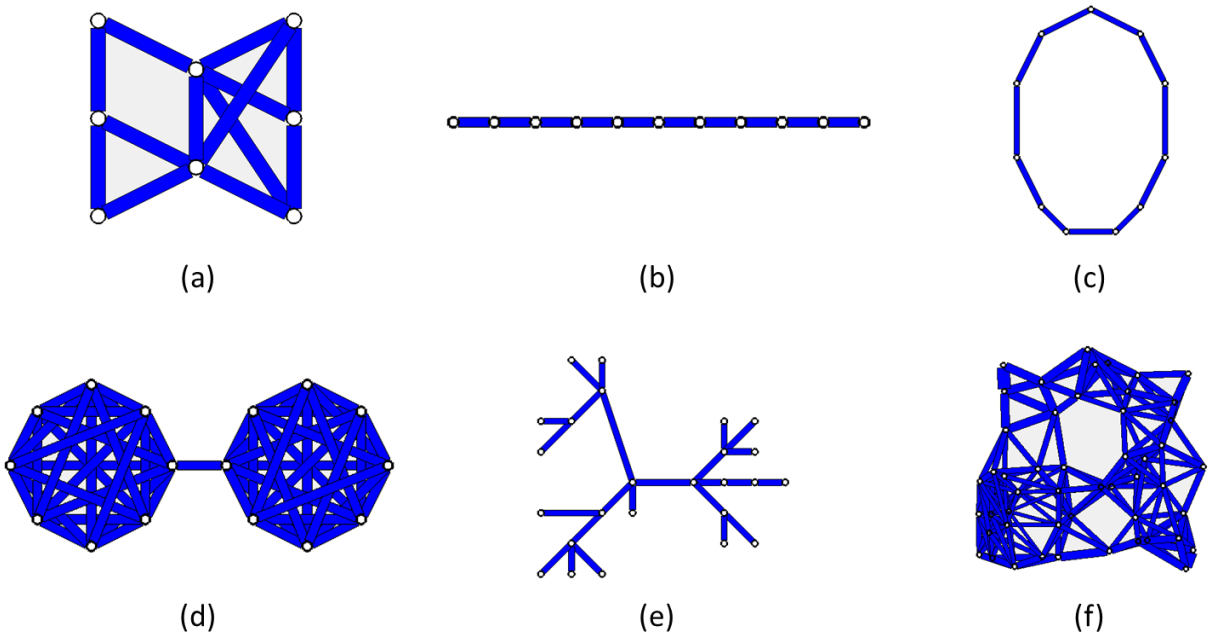

Figure 3: Patrolling environments – a) A small, 8 node graph; b) An 11 node line graph; c) An 11 node cycle graph; d) A 16 node dumbbell graph; e) A 25 node tree graph; f) A large, 50 node graph. Larger versions of these graphs can be found in Figures B-1 through B-6.

There were six primary settings that were utilized for testing in this project: a small eight node graph [10], a line, a cycle, a dumbbell, a tree, and a large randomly generated graph [10] (Figure 3). Through these graphs, we can run patrolling tests that simulate environments such as fences, perimeters, sparsely connected areas, and densely connected areas.

Patrolling Policy Generation

As mentioned earlier, we consider patrolling policies on a graph as Markov chains *M*. Given an arbitrary graph *G* and a desired patroller distribution over the graph π, we wish to generate a Markov chain with which a robot can patrol. The approach we employ is to use the Metropolis-Hastings algorithm. The Metropolis-Hastings algorithm is a sampling algorithm that approximates a probability distribution with some stationary distribution π [7]. Much like a

Markov chain, the longer the algorithm samples on the given probability distribution, the more closely the output will approximate to the desired stationary distribution.

The general form of the Metropolis-Hastings algorithm (Appendix A, Algorithm 1) generates a list of sampled values which approximates a probability distribution according to a desired stationary distribution π. As an example, say we desired to approximate a Normal distribution centered at 0 with a standard deviation of 1, $N(0, 1)$ (Figure 4). We begin the algorithm by sampling some proposal position from our current point x^i with some function $q()$ (line 4). The purpose of this proposal function $q()$ is to suggest a new position to move to based on our current position, and it can be different depending on the kind of distribution you wish to approximate. For our current example of a Normal distribution, say that we want to propose a new position by using a normal distribution centered at our current location with some standard deviation, $N(x^i, 0.05)$. Using this proposal function ($q()$: $N(x^i, 0.05)$), we acquire our proposed position to move to next. Now we can generate an acceptance probability (line 6) which determines whether we accept (line 7) or reject (line 9) the movement to this new position. The acceptance probability is the product of two ratios: the first ratio is the value of the stationary distribution at our proposed location versus our current location, and the second ratio is the probability of being at our current position given the proposed position versus the opposite. If the algorithm accepts this movement, then the next position for sampling is set to the proposed destination, otherwise we set the next position to the current position. In the case of our Normal distribution example, say we are at point A in Figure 4. If we propose point B, we can accept this movement based on the acceptance probability. Let's say that because point B is lower on the curve than point A, the movement ends up getting rejected. Now let's say that point C was sampled instead of point B.

Since point C is higher on the curve, and therefore a much more probable point to be at, the algorithm accepts this transition. After the new position is set, the process repeats for a number of iterations (line 3), more closely approximating the distribution as more points are sampled.

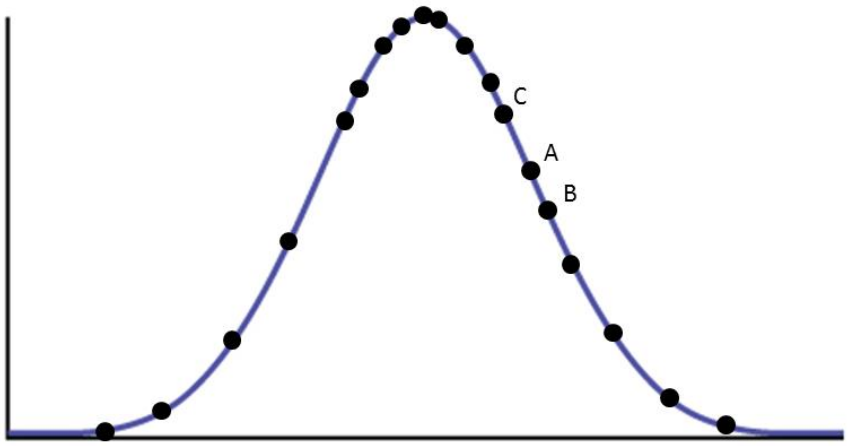

Figure 4: Metropolis-Hastings Sampling Example – This shows two possible outcomes from sampling in the Metropolis-Hastings algorithm on a Normal distribution. If the current position is point A, then we can propose a new position with the proposal function. If point B is proposed, then it is less likely to be accepted by the algorithm since it is lower on the curve compared to point A. If point C is proposed, then it is more likely to be accepted since it is higher on the curve.

Next, we present an adaptation of the Metropolis-Hastings algorithm. In their paper on finding the fastest mixing Markov chain on a graph, Boyd et al. show that the Metropolis-Hastings algorithm can be applied as a heuristic to a random walk on a graph in order to approximate a Markov chain that has the fastest mixing time with some desired stationary distribution π [8]. In this formulation, the random walk over a graph is related to the degree of the node that is the current state location (Appendix A, Algorithm 2, line 3). This also changes the acceptance probability that is utilized by the Metropolis-Hastings algorithm (Appendix A, Algorithm 3): the proposal function is now related to the degree of the graph's nodes, or the probability of a

random walk on the graph. These changes allow one to use the Metropolis-Hastings algorithm to generate a Markov chain with any desired stationary distribution π as a patrolling strategy (Appendix A, Algorithm 4).

CHAPTER III

RESULTS

When using Algorithm 4 to generate patrolling policies, there are three primary strategy types that were created: 1) approximation to a uniform distribution (lines 4-11), 2) approximation to a distribution with strong and weak clusters of nodes, and 3) approximation to a distribution with strong and weak individual nodes (lines 12-20). These three Markov chain strategies are heavily influenced by the second input of Algorithm 4, which is the desired stationary distribution π. The values of π that are being used as input are customizable depending on the type of strategy that one wishes to simulate. Some visual examples of each strategy that was used for testing on the Small Graph can be found in Figure 5. Note that if one desired to change the strategy of the patroller (strengthen/weaken different clusters/individual nodes), the respective values for each node would only need to be modified to reflect this new strategy.

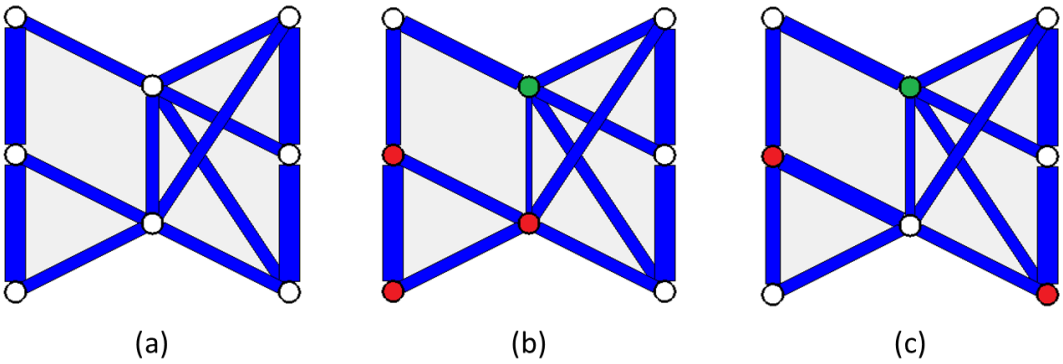

Figure 5: Three Types of Strategies – a) The Small Graph when approximated to a uniform distribution; b) The Small graph when approximated to a stationary distribution with a weak cluster of nodes; c) The Small Graph when approximated to a stationary distribution containing separated weak points. Edge width denotes the probability of travelling along that edge, green nodes denote strong nodes, and red nodes denote weak nodes. Larger versions of these graphs can be found in Figures B-7 through B-9.

Simulation Setting

Patrolling simulations are conducted in the form of a Stackleberg game [9]. Stackelberg games are two player scenarios where a leader (the patrollers) first commits to some strategy, and then a follower (the adversary) observes the leader's move and responds as optimally as possible. In our simulations, the patroller plays first by committing to a strategy that is represented by some Markov chain M with a stationary distribution π that has been approximated to one of our three strategy types. Once the robot has committed to a strategy and has begun patrolling, the adversary observes the patroller and attacks the graph for some amount of time t. Because the adversary is able to observe the patroller for an indefinite amount of time, it is assumed that he knows the stationary distribution π of the patroller's strategy. With this information, the adversary knows both the strong and weak points of the patroller's strategy and is able to watch those weak positions so as to determine the best time to attack the graph.

Results

We tested each graph a large number of times, varying the patrolling strategy, the number of patrollers, and the adversarial attack time between each set of tests. For each graph tested, the number of patrollers on the graph varied such that the maximum number of patrollers was limited to ratio of about one patroller to every eight nodes. In instances where this ratio would yield a maximum of one patroller on the graph, the maximum number of patrollers was instead raised to two so as to ensure the testing of a multi-robot patrolling scenario. The adversarial attack time in the tests was set to range from 5 time steps to about twice the number of nodes in the graph.

To begin our discussion of simulation results, we would like to focus on Figure B-15, which gives details about the testing results on the large randomly generated graph. There are a few important things to note from this graph. First, on average the stationary distribution strategy tends to return the highest chances of catching the adversary attempting to penetrate the graph while the strategy with weaker clusters tends to return the lowest chances.

Second, it is interesting to note that as the number of patrollers increase, the percentages of patroller success begins to taper off. This implies that there is a ratio of patrollers to nodes in the graph such that the saturation of patrollers on the graph will provide the highest likelihood of capturing the adversary.

Third, the rise of success between tests with different amounts of patrollers on the graph is independent of the other tests. The main difference that is noticed regarding this is that as the number of patrollers increases, the base percentage of success when adversarial attacks are quick steadily increases and the chances of success with longer adversarial attacks begin to slowly taper off. The reason for this independent rise of success rates is most likely attributed to the randomness in movement across the graph that a Markov chain utilizes. For instance, suppose we patrolled the same large graph by using a cycle instead of Markov chains. As more patrollers were introduced to the cyclic patrolling strategy, the frequency that each node would be visited would increase proportionally. Any adversarial attack time that was shorter than this frequency could be guaranteed success and anything longer would result in capture. Under this model, we would see a rise in success of capturing the adversary that is solely based on the frequency that nodes are being visited. On the other hand, since Markov chains allow the patroller to move

randomly, there is no set frequency with which a robot will visit each node. This allows a patroller to return to any node on the graph after any number of transitions, regardless of how probable it is.

The results shown for the other five types of graphs (Figures B-10 through B-14) indicate that the trends seen from the tests on the Large Graph also hold true. In almost every test case, we can see that the uniform distribution strategy still provides the highest percentages of success and the weaker clusters the worst. This disparity is more noticeable in graphs that are more sparsely connected, such as the Tree Graph, and less noticeable in denser graphs, such as the Dumbbell Graph. The tapering effect can also be seen among the other graphs, but seems to be more prominent in graphs that are more heavily connected, such as the Dumbbell Graph, and less prominent in the Line and Cycle Graphs.

CHAPTER IV

CONCLUSIONS

One of the goals of our work was to generate patrolling policies in the form of Markov chains through the use of the Metropolis-Hastings algorithm. Utilizing Boyd et al.'s adaptation of the algorithm to a random walk on a graph, we were able to create strategies based on any desired stationary distribution π, specifically the distributions that contained strong and weak spots coverage, strong and weak cluster coverage, and uniform coverage on the graph. We showed that patrolling with Markov chains that provide uniform coverage over the graph is the best strategy that patrollers can implement with this formulation. We have also showed that there is some saturation point of patrollers that should exist on the graph when performing a weighted random walk so as to give the best likelihood of capturing an adversary.

We also desired to investigate the usefulness of Markov chains as a patrolling policy representation. Indeed, Markov chains are very useful because of their non-deterministic nature. As discussed earlier, Markov chains also allow patrollers to move between nodes on a graph with some set of probabilities, which is a disadvantage for the adversary who can no longer easily predict how patrollers will move. It is also important to note that different Markov chains can yield the same stationary distribution π. This is very advantageous for patrollers because different methods can be used to create different Markov chains that conform to the same strategy type, such as uniform coverage. Such a scenario would be even worse for an adversary that is trying to penetrate a graph.

REFERENCES

[1] Portugal, D., Rocha, R. *A Survey on Multi-robot Patrolling Schemes*. In AICT 349, 2011.

[2] Elmaliach, Y., Agmon, N., Kaminka, G. *Multi-Robot Area Patrol under Frequency Constraints*. In ICRA 2007.

[3] Chevaleyre, Y. *Theoretical Analysis of the Multi-agent Patrolling Problem*. In IAT 2004.

[4] Almeida, A., Ramalho, G., Santana, H., Tedesco, P., Corruble, T., Chevaleyre, Y. *Recent Advances on Multi-Agent Patrolling*. In SBIA 2004.

[5] Agmon, N. *Multi-Robot Patrolling and Other Multi-Robot Cooperative Tasks: An Algorithmic Approach*. Diss. Bar-Ilan University, 2009.

[6] Alam, T., Edwards, M., Bobadilla, L., Shell, D. A.. *Distributed Multi-Robot Area Patrolling in Adversarial Environments.* RSN 2015.

[7] Billera, L. J., Diaconis, P., *A Geometric Interpretation of the Metrolopis-Hastings Algorithm.* In Statistics Science 2001.

[8] Boyd, S., Diaconis, P., Xiao, L. *Fastest Mixing Markov Chain on a Graph.* In SIAM 2004.

[9] Paruchuri, P., Pearce, J., Marecki, J., Tambe, M., Ordonez, F., Kraus, S. *Efficient Algorithms to Solve Bayesian Stackelberg Games for Security Applications*. In AAAI 2008.

[10] *Convex Optimization in Matlab* (Sep 2014) [Online].

Available:http://cvxr.com/cvx/examples/.

APPENDIX A

ALGORITHMS

Algorithm 1 Metropolis-Hastings(s, q(), π, N)

Input: s, q(), π, N (starting position, sampling function, desired stationary distribution of function, number of samples)

Output: vals (1xN) (N-length vector of sampled values)

1: $x^0 \leftarrow s$
2: vals ← 0
3: **for** i = 0..N-1 **do**
4: $x^* \leftarrow q(x^* \mid x^i)$
5: $u \leftarrow U_{[0..1]}$
6: **if** $u < \min\{1, \frac{\pi(x^*)}{\pi(x^i)} \frac{q(x^i \mid x^*)}{q(x^* \mid x^i)}\}$ **then**
7: $x^{i+1} \leftarrow x^*$
8: **else**
9: $x^{i+1} \leftarrow x^i$
10: **end if**
11: **end for**

Algorithm 2 pRW(A, i, j)

Input: A, i, j (NxM) (Incidence matrix of graph, graph position i, graph position j)

Output: p (probability of moving from i to j)

1: p ← 0
2: **if** (i, j) ∈ Ɛ **and** i ≠ j **then**
3: $p \leftarrow 1/d_i$
4: **else**
5: p ← 0
6: **end if**

Algorithm 3 R(A, π, i, j)

Input: A, π, i, j (NxM) (Incidence matrix of graph, stationary distribution of Markov Chain, graph position i, graph position j)

Output: p (acceptance probability)

1: $p \leftarrow \frac{\pi_j}{\pi_i} \times \frac{d_i}{d_j}$

Algorithm 4 Metropolis-Hastings-MC(A, π)

Input: A (NxM), π (Incidence matrix of graph, desired stationary distribution of Markov Chain)
Output: MC (Markov Chain)

```
MC ← 0
for i = 1..N do
    for j = 1..N do
        if π is the uniform distribution then
            if (i, j) ∈ Ɛ and i ≠ j then
                MC_ij ← min{1/d_i, 1/d_j}
            else if i = j then
                MC_ij ← ∑_{(i,k)∈ Ɛ} max{0, 1/d_i - 1/d_k}
            else
                MC_ij ← 0
            end if
        else
            if (i, j) ∈ Ɛ and i ≠ j then
                MC_ij ← pRW(A, i, j) × min{1, R(A, π, i, j)}
            else if i = j then
                MC_ij ← pRW(A, i, j) + ∑_{(i,k)∈ Ɛ} pRW(A, i, k) × (1 - min{1, R(A, π, i, k)})
            else
                MC_ij ← 0
            end if
        end if
    end for
end for
```

APPENDIX B

FIGURES

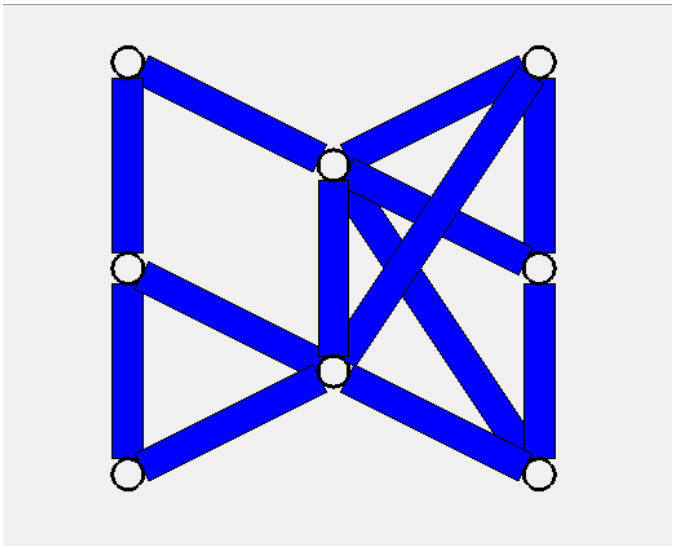

Figure B-1: Small Graph – This figure depicts a small graph consisting of 8 nodes and 13 edges.

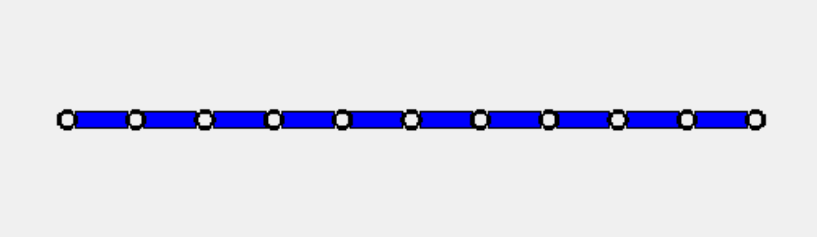

Figure B-2: Line Graph – This figure depicts a line graph consisting of 11 nodes and 10 edges.

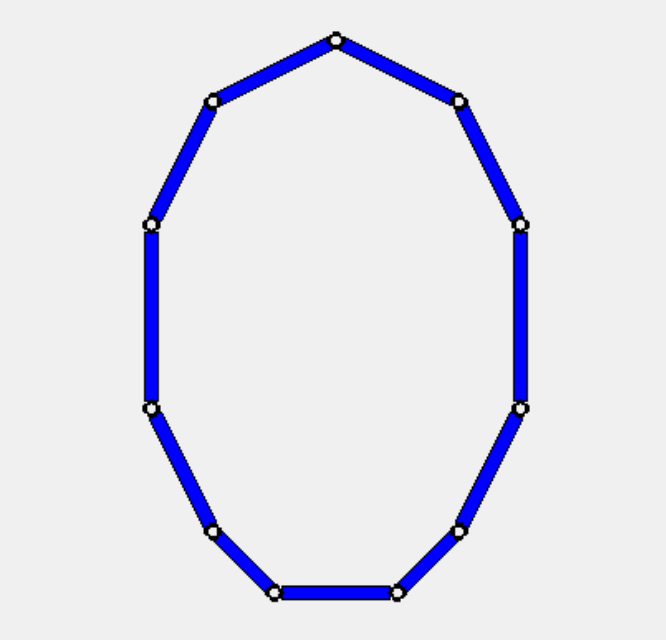

Figure B-3: Cycle Graph – This figure depicts a cycle graph consisting of 11 nodes and 10 edges.

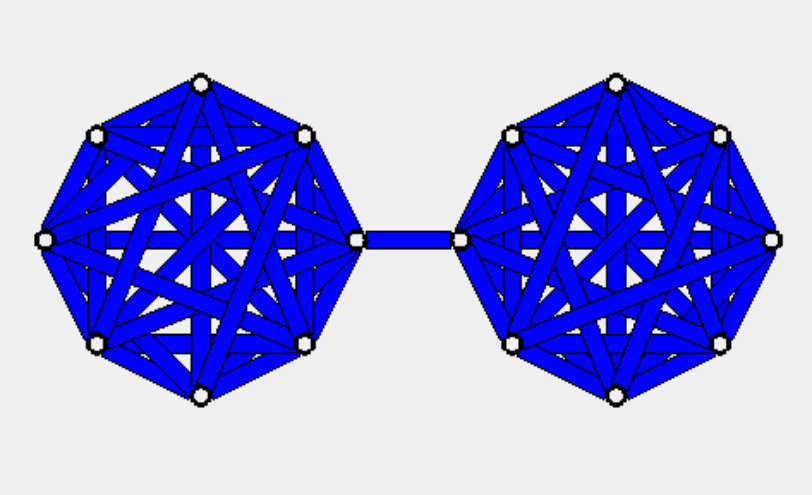

Figure B-4: Dumbbell Graph – This figure depicts a dumbbell graph consisting of 16 nodes and 56 edges.

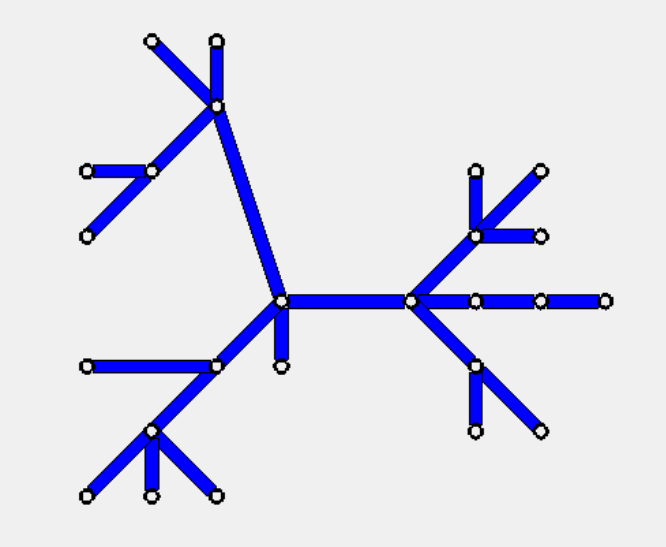

Figure B-5: Tree Graph – This figure depicts a tree graph consisting of 25 nodes and 24 edges.

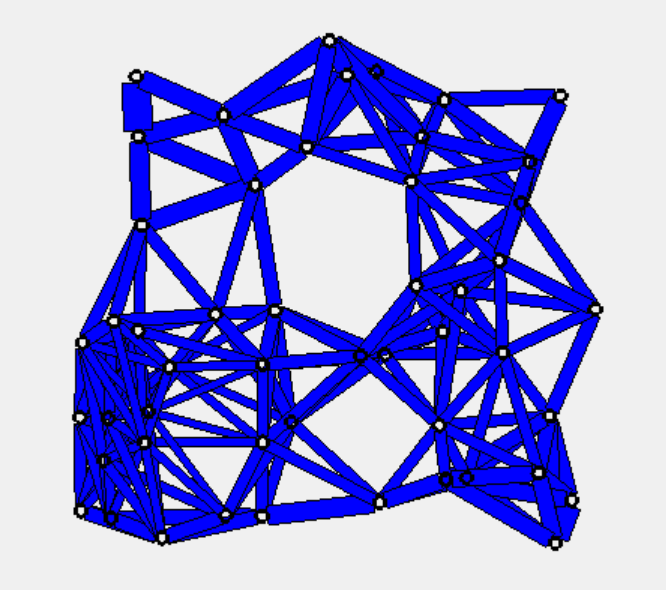

Figure B-6: Large Graph – This figure depicts a large graph consisting of 50 nodes and 200 edges.

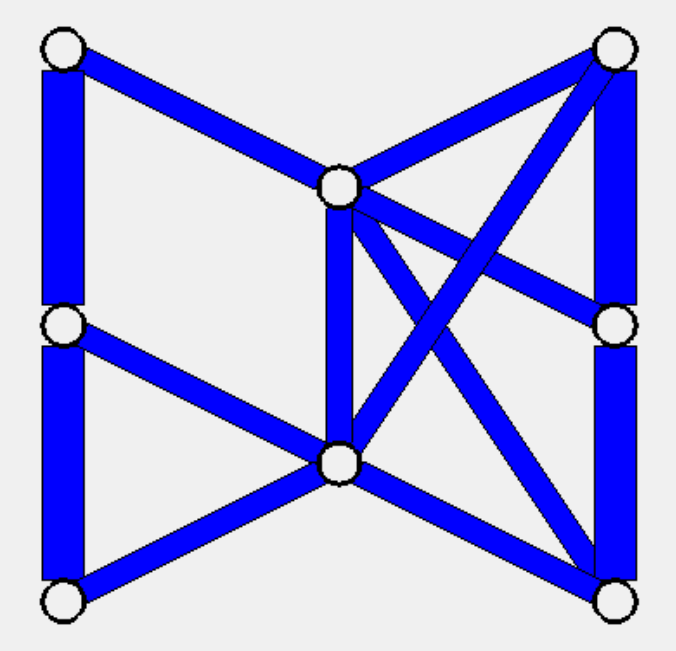

Figure B-7: Uniform Distribution Example – This figure shows the Small Graph when approximated to a uniform distribution. The width of each edge is proportional to the probability of that edge being travelled.

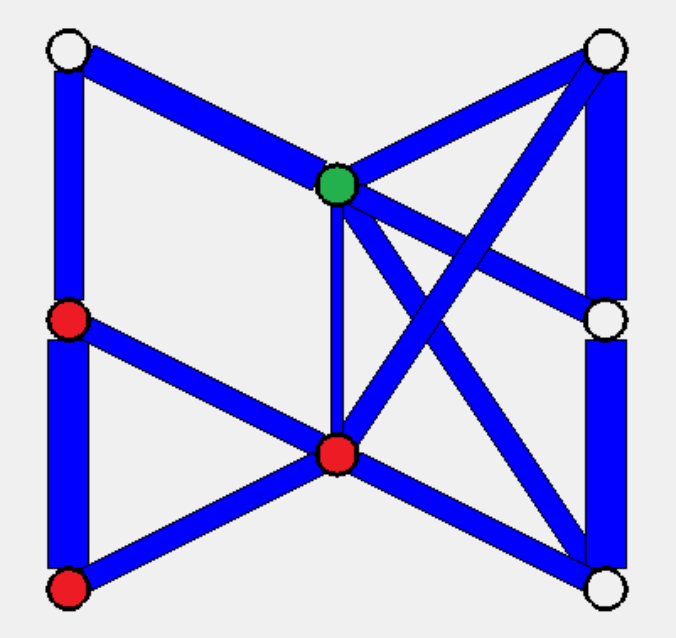

Figure B-8: Weak Cluster Distribution Example – This figure shows the Small Graph when approximated to a stationary distribution containing a weak cluster of nodes. The green node is the strong point on the graph and the red nodes are the weak nodes on the graph. The width of each edge is proportional to the probability of that edge being travelled.

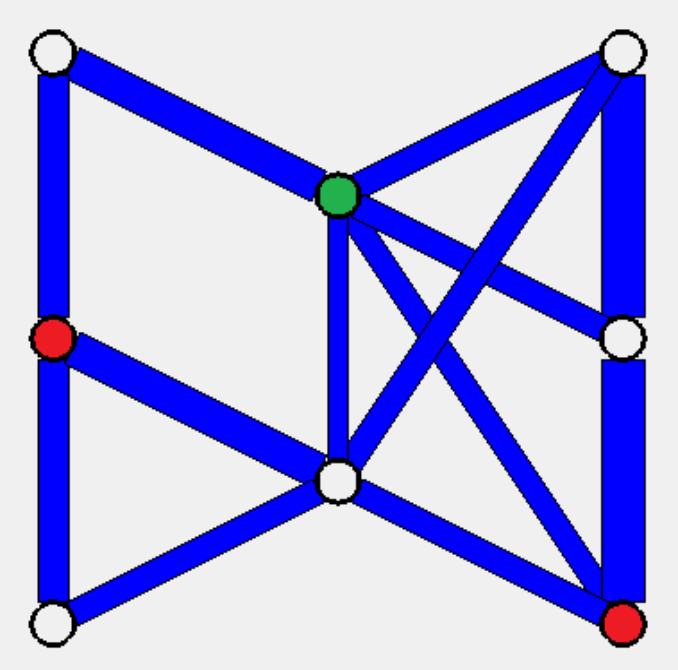

Figure B-9: Weak Points Distribution Example – This figure shows the Small Graph when approximated to a stationary distribution containing separated weak points. The green node is the strong point on the graph and the red nodes are the weak nodes on the graph. The width of each edge is proportional to the probability of that edge being travelled.

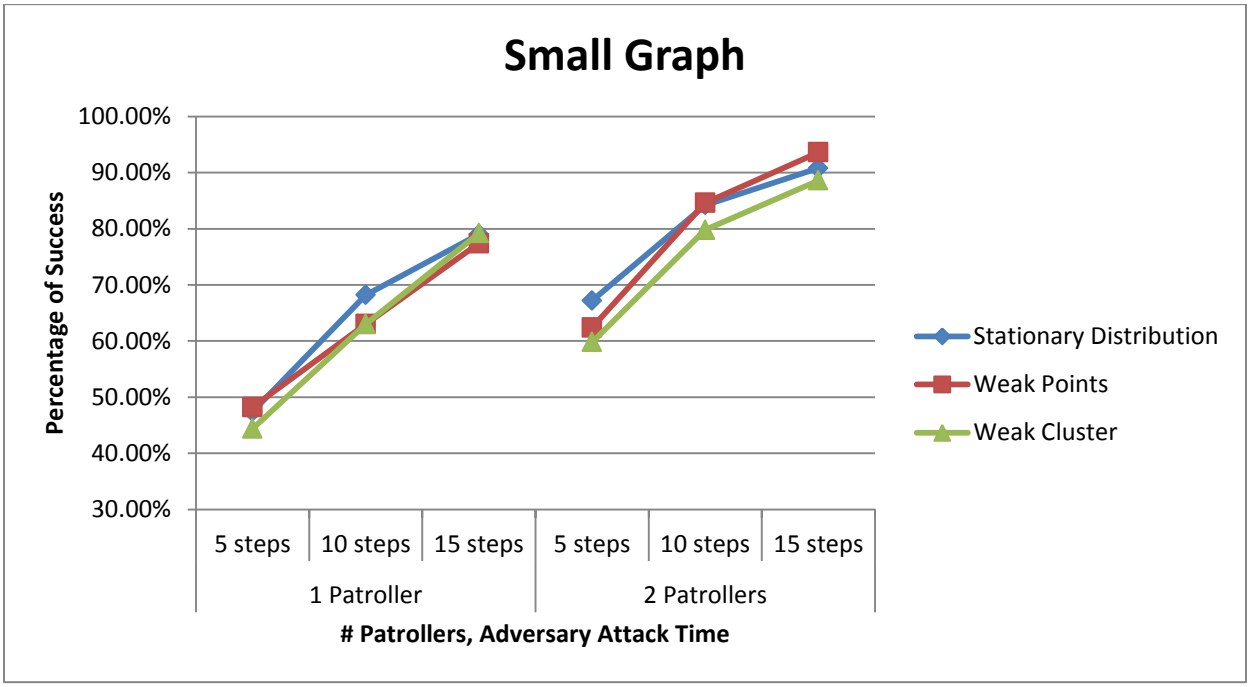

Figure B-10: Small Graph Results – This figure displays the percentage of success of detecting an adversary on the Small Graph.

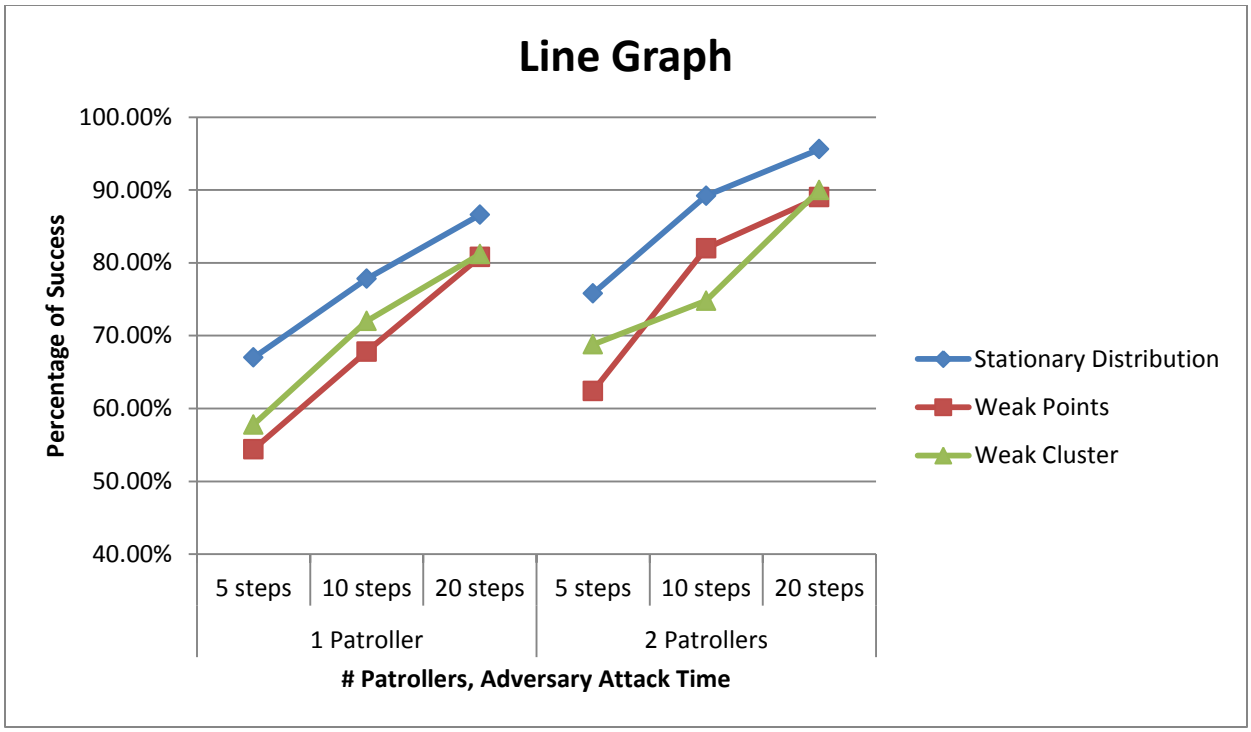

Figure B-11: Line Graph Results – This figure displays the percentage of success of detecting an adversary on the Line Graph.

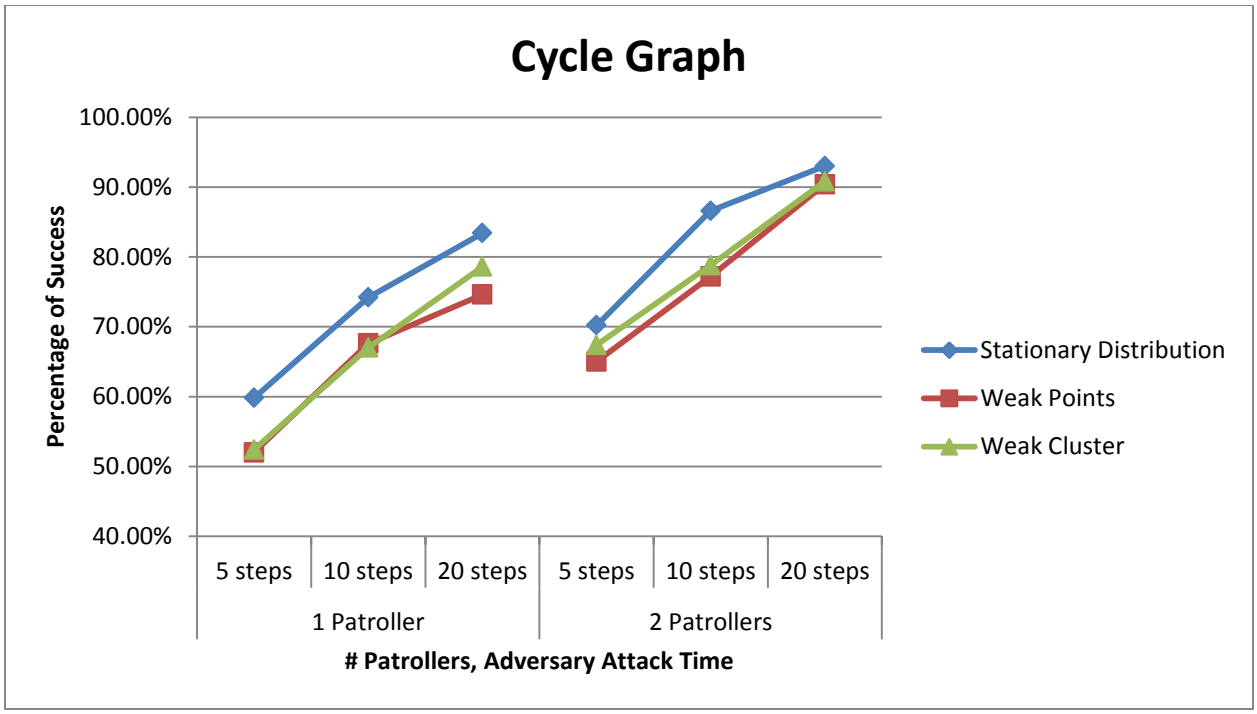

Figure B-12: Cycle Graph Results – This figure displays the percentage of success of detecting an adversary on the Cycle Graph.

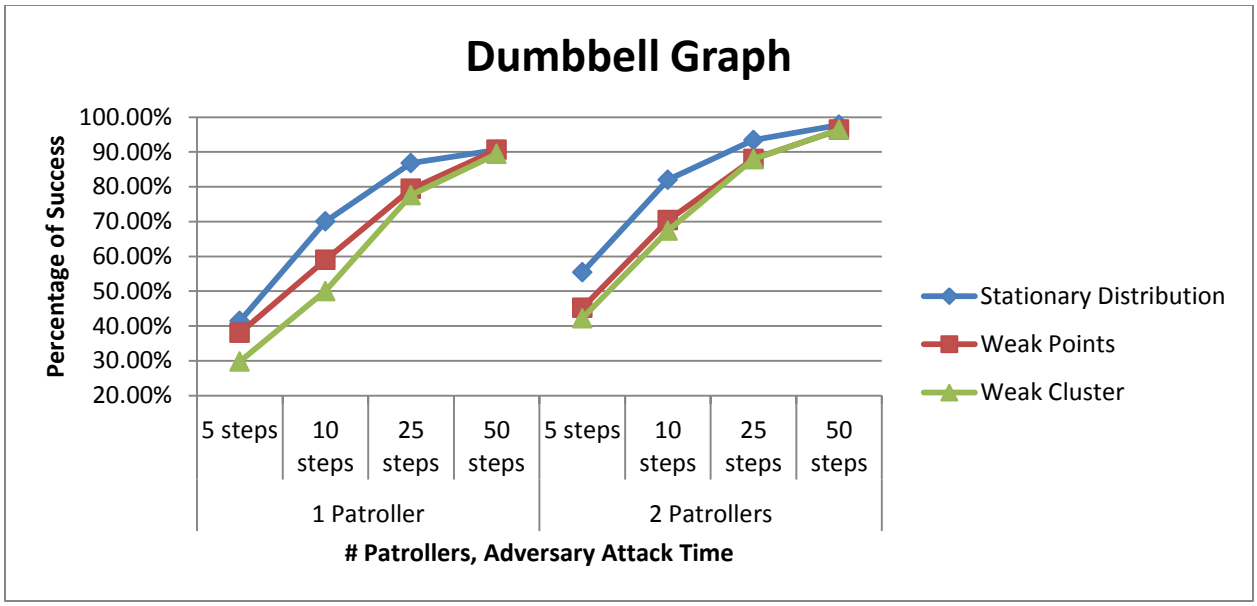

Figure B-13: Dumbbell Graph Results – This figure displays the percentage of success of detecting an adversary on the Dumbbell Graph.

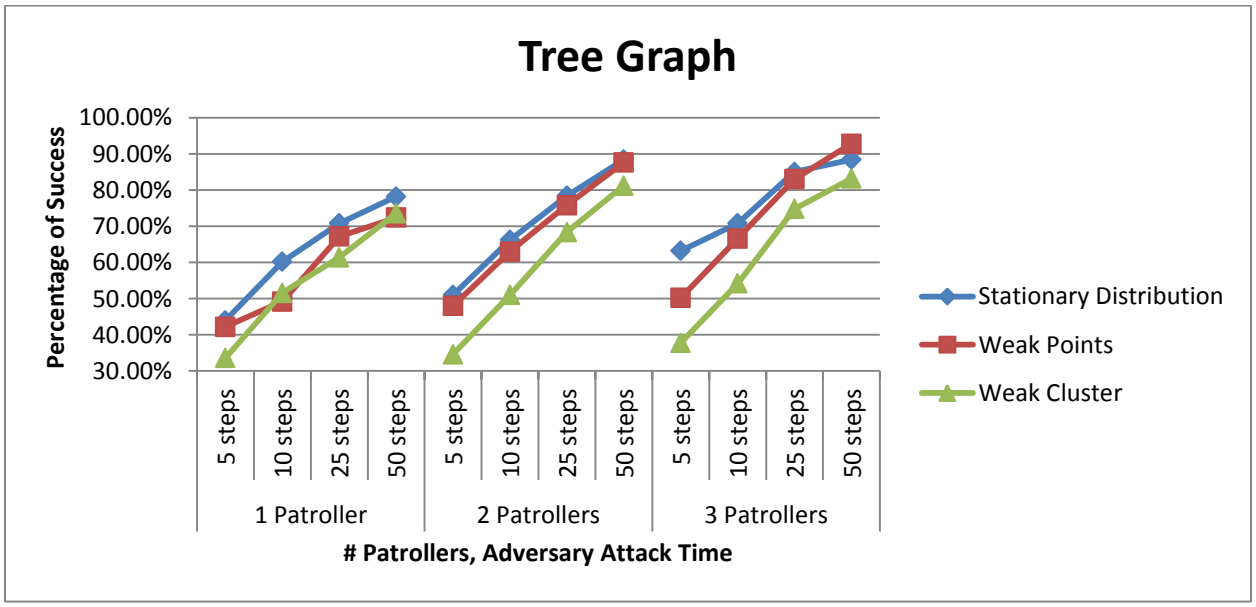

Figure B-14: Tree Graph Results – This figure displays the percentage of success of detecting an adversary on the Tree Graph.

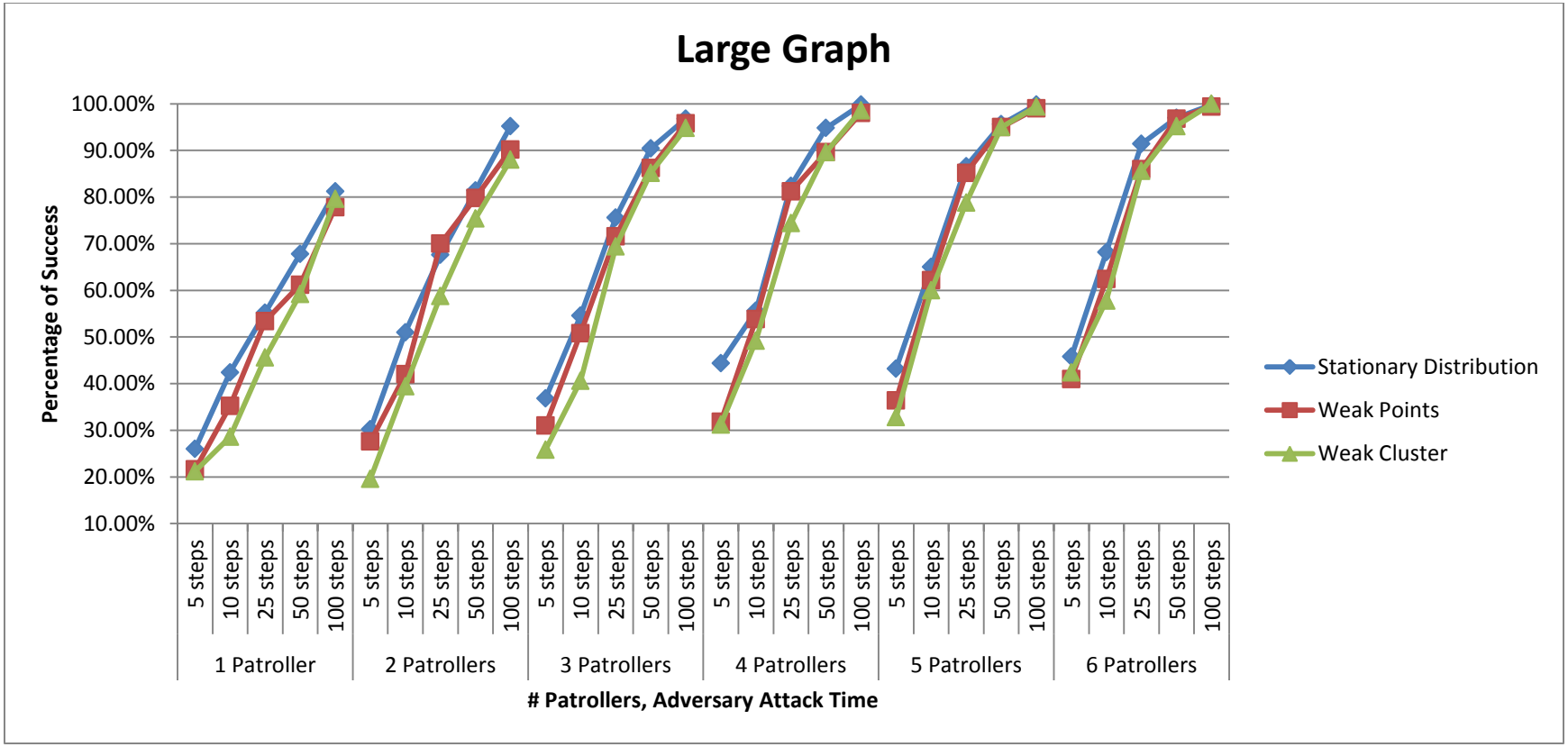

Figure B-15: Large Graph Results – This figure displays the percentage of success of detecting an adversary on the Large Graph.